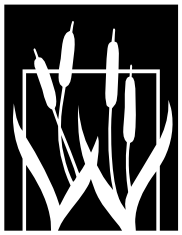

Whitemarsh
Information Systems Corporation

Return on Investment (ROI)

Data Model Manufacturing

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: Whitemarsh@wiscorp.com
Web: www.wiscorp.com

Table of Contents

Whitemarsh ROI Savings Summary 1

1.0 Issue 1

2.0 Solution Approach 3
2.1 Data Architecture Reference Model 4
2.2 Solution Approach Summary and Benefits 7

3. Solution Engineering 8

4. The ROI 10
4.1 Traditional Calculation 10
4.2 Changed Approach Calculation 11
4.3 ROI Summary 12

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

Whitemarsh ROI Savings Summary

Enterprise data models are not only at the very center of all Enterprise Business Information System environments but also enable the interrelationship among all the IT work products. Data Models are key in enabling the scope of business information systems to be restricted to just what is necessary.

Despite all the worth of Enterprise Data Models, their cost through traditional techniques has not only been excessive, they have been cited as a reason for over budget and late business information systems.

Whitemarsh's Enterprise Data Model development techniques have resolved these common complaints and also have dramatically increased the velocity through which data models are created. These techniques have produced an ROI of 8.6.

Supporting Links	
Link Area	**Link**
Enterprise Data Management Areas	http://www.wiscorp.com/roi_datamodelmanufacturingenterprisedatamanagementareas.html
The Data Administration News Letter Articles	http://www.wiscorp.com/roi_datamodelmanufacturingtdan.html
Short Papers	http://www.wiscorp.com/roi_datamodelmanufacturingshort_papers.html
Clients	http://www.wiscorp.com/roi_datamodelmanufacturingclients.html

1.0 Issue

A reasonable sized enterprise ($500 million to $1 billion annual sales), where there is a application-centric database design strategy across all its functional areas, has about 200 application-centric databases that would include involve at a very minimum:

- Customer Information
- Sales and ordering Information
- Marketing Information
- Human Resources Information
- Finance (GL, AP, AR, etc) Information
- Manufacturing, Inventory Information
- Warehouse Information
- Distribution Information

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

- Middle management Information
- Senior Management Information
- Strategic, Tactical and Operational Planning

The common overlap among these data models is about 20 percent. That is, about 20 of every 100 tables is somewhat to exactly the same as tables in one or more other databases. Because of these duplications,"data harmony" can exist only when all the following statements are completely true individually and in unison:

- All the columns of the duplicated tables are exactly alike, and
- The granularity of the duplicated table rows are exactly the same, and
- The business events or time slices when the data is re-created/re-collected are exactly the same, and
- The semantic meanings, definitions, data types and value domains are exactly alike.

Alternatively, if any or all the "data harmony" statements above are false, the problems associated with data dis-integration though tedious, error-prone, fragile, and expensive can be managed. Collectively, successful data dis-integration management is called "data governance." Otherwise it is just called "chaos."

There are three solutions employed for these kinds of data overlaps/duplication. The first is to re-collect/re-create the data in each of these common tables such that all the "data harmony" statements are true, and the second is to create an ETL (Extract-Transform-Load) application system environment that extracts the data from one database, transforms it, and loads it into another. Both solutions have real and obvious challenges.

The first solution, re-collection and re-creation, requires that every data collection system has to be modified to continuously store data in multiple application databases whenever that data is either collected or updated. Such application efforts cause application information system bloat, increased architecture, design, implementation, testing, and maintenance. Such systems require continuous evolution as there are ever more databases to deal with.

The second solution is ETL. That is, Extract-Transform-Load. These are a kind of business information systems in their own right, one for each different instance of the process of Extracting, Transforming, and Loading. Companies abound that address ETL with software packages/systems that regularize, standardize, and coordinate all the executions of the various ETL processes throughout the enterprise.

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

To give dimension to this data interchange problem, consider that if there are the 200 application-centric databases, and there is a need to share data across 20% of them. The quantity of interfaces for either of either solution is computed to be:

$$(N * (N-1)/2) * 0.2$$

In this case, N = 200. The quantity of data-exchange interfaces is thus, 3,980.

The first alternative, re-collection and re-creation requires that each business information system touched by one or more of these 3980 interfaces has to be modified to get the redundant data from an authoritative source business information system and store it into the interfaced system. Each such business information system modification becomes a mini-maintenance cycle that requires its own requirements through user-acceptance cycle of phases. If the interface affects a single table, the cost of the modification is $16,000. That's 80 function points for the table with a per function point cost of $200. Over the 3880 such mini-business information system modifications, the overall cost is $63 million.

Under the ETL alternative, if each ETL interface need requires only ONE staff week to establish an ETL process through the application of a very sophisticated ETL tool set (e.g., requirements, architecture, design, construction, testing, and documentation), and then, over 5 years, another staff week to maintain it, the cost is almost $32 million. That represents the full time staff of about 30 (3980 interfaces * 80 hours per interface / 5 years / 2080 staff hours per year).

Regardless of the increased efficiencies from either the re-collection/re-creation solution or the ETL solution, the issue is very large, and the solution very expensive.

The third solution is the most beneficial solution. It is to not have the problem in the first place. That requires an overall data architecture, and prior to that a data architecture reference model that enables the problem to either never be created in the first place, or that provides a pathway to eliminate the problem altogether over time.

2.0 Solution Approach

The accomplishment of the solution to enterprise-wide data sharing involves the creation and implementation and evolution of a data architecture reference model, and the use of the created data architecture reference model as the implementing mechanism through which databases are created, employed, and evolved.

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

The data architecture reference model itself is accomplished top-down. The use of the data architecture reference model as the database implementing mechanism can be either top-down, or bottom-up through reverse-engineering.

2.1 Data Architecture Reference Model

The Data Architecture Reference Model consists of the following contained major models:

- Data Element Models
- Concepts Data Model
- Database Logical Model
- Database Physical Model

Data Element Model. Data elements are the definition and stated semantics for enterprise business facts employed as the semantic foundations for attributes of entities within data models of concepts, columns of tables within database logical models that support business requirements, and ultimately the DBMS columns within database physical models that support the actual business operations. An example of a data element is Invoice Number.

Data elements are not just miraculously discovered. Rather they are derived from enterprise concepts represented through value-based expressions. The value-based expressions are founded on conceptual value domains, and are finally detailed through actual value domain value-sets.

In the case of the data element Invoice Number, the highest level concept might be Finance, and within that might be Instrument and within that might be Liability. As to the type of data, that is, Number, the Conceptual Value Domain would be Numeric with a Value Domain of positive integers, and a set of value domain values from 0001 through 2 billion.

Because of the Data Element Model work, all business fact representations in the concepts data model, and the database logical and physical models can be standardized. Additionally, because data elements are set within their encapsulating concepts and conceptual value domains, the data elements can be seen to be semantically similar even when their names are different. These standardizations dramatically affect–in a positive way--the “data harmony” bullets above.

Concepts Data Model. Concept Data Models are data models based containers of concepts. Examples are students, schools, organizations, addresses, finances, inventory, locations, and the like. These concept containers are deployed as multi-business fact semantic templates within one or more tables within one or more Database Logical Models.

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

Every concept data model entity attribute should map to its data element semantic parent. Concept data model entity attributes can also have assigned semantic and data use modifiers that further restrict the meaning and value domains of an attribute that may exist within the scope of the attribute's parent data element. Finally, Relationships can interrelate entities within and across multiple subjects.

An example of a concept model is Invoices. In this example, the subject is Invoice, and of its contained entities two are Invoice Header and Invoice Detail. The attributes for each of these entities would be picked from the list of available data elements. Once a data element is picked, its propertiesare automatically copied and transformed into the appropriate properties of the entity's attribute. The primary key for each is InvoiceHeaderPKey and InvoiceDetailelPkey. Because of the previously created attributes, the appropriate ones for each entity are picked. A foreign key, that is, the backward relationship from Invoice Detail to Invoice Header would be established by replicating the Invoice Header's primary key attribute as an attribute within the Invoice Detail entity.

Because of the Concepts Data Model, the construction of all business fact representations in the concepts data model entities, that is, the concept data model entity attributes are all mapped back to their data semantic source, the data element.

This enables, the standardization of semantics even when the names of the data elements and the concept data model entity attributes are different. After all, "a rose by any other name is still a rose." This standardization dramatically affects even more of the "data harmony" bullets above.

Database Logical Model. Database Logical Models, are the data models of databases that are independent of database management systems (DBMS) such as Oracle, DB2, MS/SQL and Sybase. A database is bounded by a schema which, in turn, contains tables. Within tables are columns, primary keys, foreign keys, and unique keys. Relationships are restricted to tables within a single schema.

The point of having a database logical model is to specify database tables within the scope of a single schema that are completely independent of any consideration of performance.

Each Database Logical Model can import multiple collections of concept data model entities from multiple subjects to satisfy the overall requirements of the Database Logical Model schema. It can also be appropriate to import Concept Data Model entities for use in multiple Database Logical Model tables.

Every Database Logical Model table column should map to a parent Attribute from an entity of a Concept Data Model. Semantic and data use modifiers can be assigned to every column that can

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

further restrict the meaning and value domains of a column that may exist within the scope of the column's parent attribute.

An example of a Database Logical Model is Financial Management. Here, the schema is Finance. Its table collections might include vendors, customers, accounts, cost centers, bank accounts, budgets, exchange rates, assets, depreciations, and the various transaction specifications for each table collection, including for example, InvoiceHeader and InvoiceDetail. The columns within each of these tables are automatically generated through the selection of the Concept Data Model. Created including column names, data types and the automatic mapping back to attributes from the appropriate Concept Data Model entity attributes.

The primary key for Database Logical Model table is created and because of previously created columns, the one appropriate set for each table is picked. A foreign key, that is, the backward relationship from InvoiceDetail to InvoiceHeader is established by replicating the InvoiceHeader's primary key column as a column within the Invoice Detail table.

Because of the previous work on the Concepts Data Model and the Data Element Models, the effort required to create a Database Logical Model is dramatically reduced. Further, this process enables the standardization of semantics even when the Database Logical Model table column names are different. This standardization is the culmination of positive effects on the "data harmony" bullets above.

Database Physical Model. Database Physical Models, are the data models of databases that are dependent of database management systems (DBMS) such as Oracle, DB2, MS/SQL and Sybase. A database is bounded by a DBMS schema which, in turn contains DBMS tables. Within DBMS tables are DBMS columns, DBMS primary keys, DBMS foreign keys, DBMS unique keys and DBMS secondary keys (non-unique values). Relationships are restricted to DBMS tables within a single DBMS schema.

The point of having a Database Physical Model is to specify DBMS tables within the scope of a single DBMS schema that are either tuned to efficient update or are tuned for efficient reporting. When tables are tuned for update they are commonly in third-normal form. When tables are tuned for reporting, they are commonly denormalized.

Each Database Physical Model can import multiple collections of Database Logical Model tables from multiple Database Logical Models to fulfill the overall requirements of the Database Physical Model schema. It can also be appropriate to import Database Logical Model tables for use in multiple Database Physical Model DBMS tables.

Every Database Physical Model table column should map to a parent column from a table of a Database Logical Model.

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

One example of a Database Physical Model is Invoicing. This database might be a subset of the Database Logical Model, Finance that has been created to specially serve the needs of a Invoicing business information system. Within this special database there could be the InvoiceHeader and InvoiceDetail DBMS tables. The DBMS columns within each of these DBMS tables are automatically generated through the selection of the Database Logical Model. Created are the DBMS column names, DBMS data types and the automatic mapping back to columns from the appropriate Database Logical Model table columns.

An opposite sized database example would be a very large, multiple-functional-area Database Physical Model with thousands of DBMS tables that are brought together from multiple Database Logical Models. In such a database there can be multiple collections of highly connected tables and intersection tables that relate tables from the different Database Logical Models to each other.

The first alternative, the subset database approach, closely tracks with the application centric database approach that are highly cohesive but very loosely coupled. Such a collection of databases would ultimately give rise to the need for the two data sharing problems described at the start of this ROI. This second alterative, a database of thousands of tables could be such that much of the two shared data problems are designed out of existence. This elimination would occur if the Data Architecture Reference Model is followed from Data Element to Concepts Data Models to Database Logical Models to Database Physical Models. It will be through these levels that maximum“data harmonization” can be accomplished.

2.2 Solution Approach Summary and Benefits

The solution approach is summarized as follows:

Data Element Model. The Data Element Model with all business fact representations represents the foundation for data semantics management of the attributes in the Concepts Data Model, the Columns in the Database Logical and Physical models.

Additionally, because data elements are set within their encapsulating concepts and conceptual value domains, the data elements can be seen to be semantically similar even when their names are different. These standardizations dramatically affect–in a positive way--the “data harmony” bullets above. The relationship between data elements and attributes is one-to-many.

Concept Data Models. The Concepts Data Model enables the construction of containers of multi-business-fact data structure templates that both map backwards to parent Data Elements from the Data Element Model and forward to one or more tables within one or more Database

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

Logical Models. The relationship between Concepts Data Model and Database Logical Model is many-to-many.

Because of the backwards mapping from attribute to data element, the names of data elements and those of attributes can be different. Additionally, attributes can be constructed such that their semantic and data use modifiers can be more restrictive subsets from those specified for data elements.

Database Logical Models. The Database Logical Models are quickly and easily constructed by imports from the Concepts Data Models. The process is not one of invention. Rather, it is one of maximum reuse.

The Database Logical Model manufacturing process enables the standardization of semantics even when the Database Logical Model table column names are different. Additionally, columns can be constructed such that their semantic and data use modifiers can be more restrictive subsets from those specified for attributes. This standardization is the culmination of positive effects on the "data harmony" bullets above.

Database Physical Models. Database Physical Models, like Database Logical Models are quickly manufactured through imports from the Database Logical Models. Again, because of the backwards mapping process, DBMS Column names can be different from column names even though their other semantics are the same.

Database Physical Models can be of two extremes or a mixture of both. The first extreme is the subset database approach which closely tracks with the application centric database approach that are highly cohesive but very loosely coupled. Such a collection of databases would ultimately give rise to the need for the two data sharing problems described at the start of this ROI.

The other extreme is the expansive database that contains thousands of tables that enable elimination of much of the two shared data problems described at the start of this ROI. Elimination, however, requires following the Data Architecture Reference Model from Data Element to Concepts Data Models to Database Logical Models to Database Physical Models. It will be through these levels that maximum"data harmonization" can be accomplished.

3. Solution Engineering

The engineered solution is founded on collections of highly engineered data models within the Metabase System.

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

- Data Element Models
- Concepts Data Model
- Database Logical Model
- Database Physical Model

Data Element Model. Data elements consist of a number of tables for storing data elements including the data elements encapsulating business environment of Concepts, Conceptual Value Domain, Data Element Concept and Data Element Classifications. Each data element is supplemented with Value Domains. Semantic and Data Use Modifiers and data that can be assigned to each data element concept and data element to modify the data element's meaning and use. There is a one-to-many relationship between data element and attribute.

Concepts Data Model. Concept Data Models consist of number of Metabase System tables for storing subjects, entities, attributes, and inter-entity relationships. Relationships can interrelate entities within multiple subjects. Every entity attribute should map to its parent data element. Semantic and data use modifiers can be assigned to entity-attributes. There is a one-to-many relationship between attribute and column, and there is also the ability for an entity to be semantically deployed across many tables, and multiple entities to be deployed within one table.

Database Logical Model. Database Logical Models consist of number of Metabase System tables for storing the data structure components: schema, tables, columns, and inter-table relationships. Relationships are restricted to tables within a single schema. Every table column should map to its parent attribute. Semantic and data use modifiers can be assigned to every table column. There is a one-to-many relationship between column and DBMS column, and there is also the ability for one table to be employed across many DBMS tables, and multiple tables to be employed within one DBMS table.

Database Physical Model. Operational Database Models consist of number of Metabase System tables for storing: DBMS schema, DBMS tables, DBMS columns, and inter-table DBMS relationships. Relationships are restricted to DBMS tables within a single DBMS schema.

DBMS Relationships are restricted to DBMS tables within a single DBMS schema. Each Database Physical Model can address multiple Implemented Database Models. There is a backwards relationship from every DBMS column to its parent column. As stated above, there is also the ability for one DBMS table to be deployed from multiple tables, and multiple DBMS tables to be deployed from one table.

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

4. The ROI

4.1 Traditional Calculation

The analysis, design, and construction effort assumes a journeyman data modeler and a subject matter expert for each functional area represented in the database's design. A database table consists of fundamental components, which are:

- Table specification
- Column specifications
- Primary key specification
- Foreign key specifications
- Unique key specifications
- Secondary key specifications

An average data model for a application-centric database and single business information system is about:

- 100 Database Schema Tables
- 10 Columns
- 1 Primary Key
- 1 Unique Key (based on business value columns)
- 1.5 Foreign Keys
- 2 Secondary Keys

The design and construction times for these components are:

- Table: 2 hours.
- Column: 30 minutes.
- Primary Key: 30 minutes
- Unique Key: 30 minutes
- Foreign Key: 30 minutes
- Secondary Key: 30 minutes

Thus the overall design and construction time for a typical 100 table database is:

- 200 hours for the tables
- 500 hours for the columns.
- 50 hours for Primary Keys
- 50 hours for Unique Keys
- 75 hours for Foreign Keys

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

- 100 hours for Secondary Keys

Total for each application centric database: 975 staff hours. The total enterprise staff hours (200 databases): 195,000 (94 staff years). At a cost of $100 per staff hour, the overall cost for all 200 databases is $19.5 million.

4.2 Changed Approach Calculation

The design and construction times for these components are based on the previously developed Data Element Model and Concepts Data Model. The typical database as above are:

- 100 Database Schema Tables
- 10 Columns
- 1 Primary Key
- 1 Unique Key (based on business value columns)
- 1.5 Foreign Keys
- 2 Secondary Keys

This makes the ROI computation equivalent between a traditionally developed database and a Database Logical Model:

- Table: Selection and importing: 5 minutes for each table.
- Column: Importation, review, and modification: 30 minutes per table.
- Primary Key: 10 minutes
- Unique Key: 10 minutes
- Foreign Key: 10 minutes
- Secondary Key: 10 minutes

Thus the overall design and construction time for a typical database with quantities above is:

- 8.3 hours
- 50 hours for the columns.
- 16.6 hours for Primary Keys
- 16.6 hours for Unique Keys
- 16.6 hours for Foreign Keys
- 16.6 hours for Secondary Keys

Total for each application centric database, the changed approach hours is: 141.3 staff hours. The total enterprise staff hours (200 databases): 28,260 hours (13.5 staff years). At a cost of $100 per

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved

staff hour, the cost for a single database would be $141,300. The overall cost for all 200 databases is $2.826 million.

Now, because the application of the Data Architecture Reference Model enables the elimination of the 20% overlap, the price for the database would likely be reduced from 141.3 hours to 113 hours and across the enterprise from 28,260 hours to 22,608 hours.

From a cost point of view, the reductions for a single database is reduced from $141,300 to $113,040 and for all 200 databases to $2.261 million.

4.3 ROI Summary

The bottom line of the ROI is simply this:

	Single Database		**200 Databases**	
Approach	**Hours**	**Cost**	**Hours**	**Cost**
Traditional	975	97,500	195,000	19.5 Million
Changed	113	11,300	28,260	2.26 Million
ROI	8.6			

Copyright 2015, Whitemarsh Information Systems Corporation
Proprietary Data, All Rights Reserved